

A SIMPLE HIGH SPEED BIPOLAR MICROPROCESSOR ILLUSTRATES SYSTEM DESIGN AND MICROPROGRAM TECHNIQUES

Prepared by:

Bill Blood

Applications Engineering

High speed bipolar LSI 4-bit slice circuits can significantly reduce processor system package count. This is shown with a microprocessor which uses only 10 integrated circuit packages to perform 2's complement add, subtract, multiply and divide.



MOTOROLA Semiconductor Products Inc.

INTRODUCTION

The engineer familiar with MOS microprocessors learns new skills when designing a higher performance bipolar LSI system. Bipolar LSI 4-bit slice circuits are building blocks allowing the designer to configure a processor architecture, size, and instruction set for optimum performance. Additional flexibility is gained through the use of microprogramming because the processor can perform more complicated instructions such as multiply and divide or, alternately, can be designed to take advantage of existing software.

The following text goes through a complete bipolar LSI system design. Steps include define the system, block out system sections, set up the microprogram structure, pick bipolar LSI parts, generate a microprogram, and finalize the system design. Three system goals are used throughout the project.

1. Maximize the use of bipolar LSI.
2. Show the flexibility of microprogramming.
3. Maintain the bipolar LSI speed advantage.

The system is being designed as a demonstration and technical support tool. However, the design flow and decisions involved represent a wide range of bipolar LSI processor systems.

DEFINE THE SYSTEM

System functional requirements must be defined prior to hardware design. This seems obvious, but changes later in the design cycle complicate programming and often increase package count. Figure 1 shows the I/O structure and program functions of this processor project.

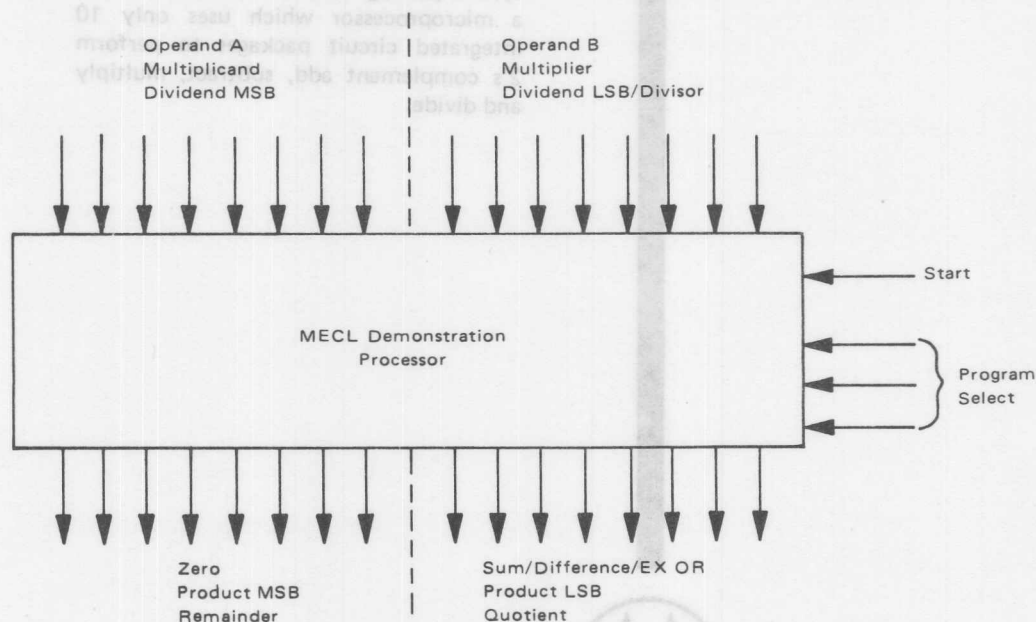


FIGURE 1 — I/O Description

The system has 16 input lines which enter a 16-bit data word or two 8-bit words in parallel. Similarly, 16 output lines display the results as either one 16-bit word or two 8-bit words.

A program select input port selects which processor program is executed.

1. ADD, SUBTRACT, and EXCLUSIVE OR read two 8-bit operand inputs and generate an 8-bit answer. The eight most significant bits are held at zero.
2. MULTIPLY reads an 8-bit multiplicand and 8-bit multiplier. The answer is a 16-bit product.
3. DIVIDE requires two data inputs. First a 16-bit dividend is entered, then the 8-bit divisor. The answer is an 8-bit quotient and 8-bit remainder.

The final processor input is a start signal given after input lines and program select are present. Since start could be from a pushbutton or toggle switch, bounce elimination and pulse shaping are included in the system design.

BLOCK OUT THE SYSTEM

The major blocks of a bipolar LSI processor are ALU, I/O, microprogram sequencing control, and microprogram memory. These functions are interconnected as shown in Figure 2. The ALU block handles all arithmetic, logic, and shift operations. It also includes working registers for temporary storage and a means for the input of new data and output of results. The MICROPROGRAM SEQUENCING block generates the microprogram memory address and provides a method for sequencing through microprogram. It combines with BRANCH LOGIC to make tests for conditional jumps in program.

Circuit diagrams external to Motorola products are included as a means of illustrating typical semiconductor applications; consequently, complete information sufficient for construction purposes is not necessarily given. The information in this Application Note has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described any license under the patent rights of Motorola Inc. or others.

The key to building a bipolar LSI system is in partitioning the MICROPROGRAM MEMORY block. The memory is divided into sections called fields, each capable of an independent function. Any combination of these fields can be selected to execute a microinstruction.

In Figure 2, the PROGRAM FLOW field is separated into two parts. The INSTRUCTION section tells the microprogram sequencing logic how to generate the next program address. Instructions include increment, jump, branch on condition, jump to and from subroutine, etc. The NEXT ADDRESS section of the program flow field provides a destination address for direct jump or conditional branch instructions.

The BRANCH field sets up test parameters for conditional jumps in program. For example, a jump if ALU results equal zero would be accomplished by gating zero detect from the ALU to a branch input on the microprogram sequencer. The corresponding INSTRUCTION would be branch on condition and NEXT ADDRESS contains the conditional branch destination. In this manner it is possible to select any number or combination of test signals.

DATA and ALU fields go to the ALU and I/O block. The DATA field controls data transfers between the input ports, output ports, and internal working registers. The ALU field controls the various arithmetic, logic, and shift operations required to execute a program.

The system being developed here is designed around three LSI circuits as shown in Figure 2. Two MC10803s handle all ALU operations, provide working registers, and control the 16 input and 16 output lines. A single MC10801 provides logic for microprogram flow and addressing. Additional MC10801 and MC10803 information is given in the following key LSI parts section.

Processor programs are stored in the microprogram memory. This system uses four 10139 PROMs in a 32-word by 32-bit organization. At first it may seem surprising that add, subtract, multiply, divide, and exclusive OR are all stored in 32 memory words. However, each word is 32 bits wide so several processor functions can be performed in parallel. These seven integrated circuits form the processor nucleus. Three additional SSI parts provide for crystal oscillator, power-up reset, and start pulse shaping. Branch logic gates route test signals into the MC10801 for conditional jumps in program. The result is a high-speed bipolar processor with only ten integrated circuit packages.

Figure 2 can be used to define the term microinstruction. A microinstruction executes all functions in a microprogram memory word. A clock signal into the MC10801 microprogram sequencer logic puts a program address location on the address lines. The microprogram memory then sets up select lines on the MC10803 to read data in, operate on the data, and display or

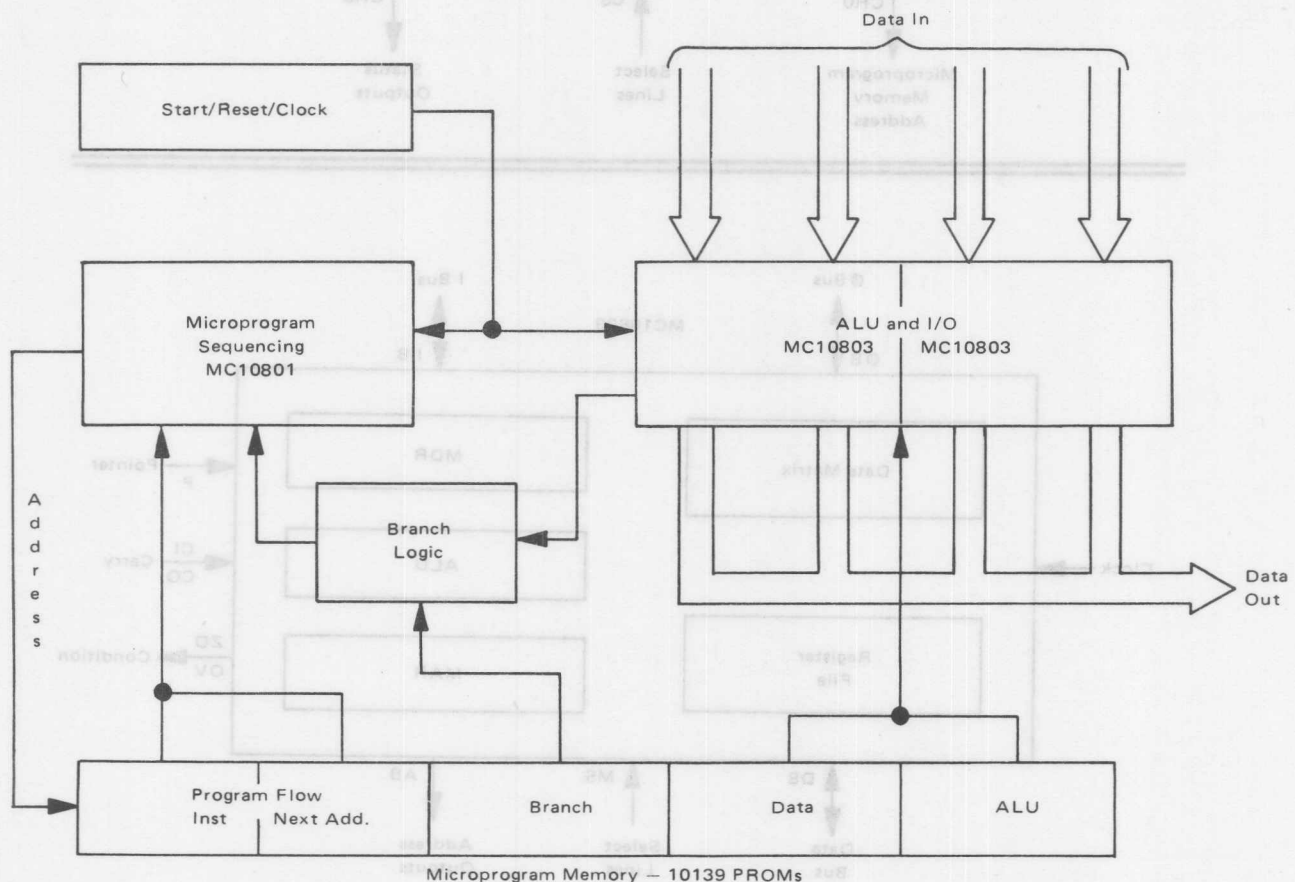


FIGURE 2 - Processor Block Diagram

store the results. In parallel with the ALU, a new microprogram address is being generated from microprogram memory and branch logic inputs to the MC10801. A second clock pulse gives the microprogram a new word address and clocks any ALU function into the appropriate storage register. System performance is measured by the microinstruction time and by the number of microinstructions required to execute a program.

The flexibility of microprogramming allows the

processor to perform a wide variety of system functions. The five programs—add, subtract, exclusive OR, multiply, and divide—being developed are only a small sample of the possible combinations. The processor could be expanded for process control, data formatting, digital filtering, minicomputer design, peripheral controllers, etc. Four MC10803s will directly operate on 16-bit words and give a corresponding increase in I/O lines. Two MC10801s allow microprogram memory expansion to 4K words for more comprehensive programming.

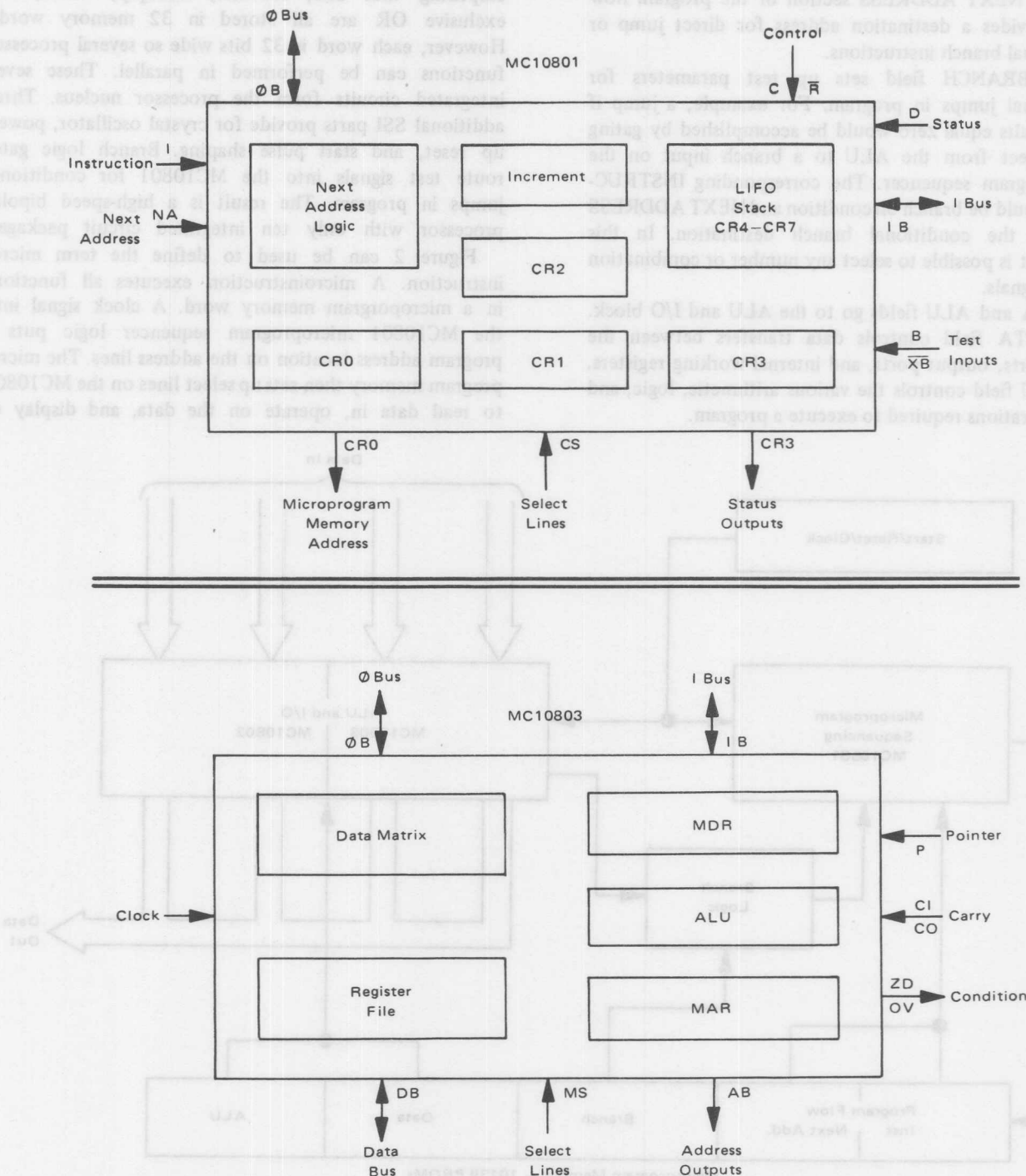


FIGURE 3 — Key LSI Parts

KEY LSI PARTS

Prior to continuing it is important to look at the MC10801 and MC10803 LSI parts. The various internal sections of each part are shown in Figure 3. MC10801 CR0 register holds the microprogram memory address. Sequencing information goes into the Next Address Logic block where it is decoded and the correct next address routed to CR0. An incrementer is used with several sequencing commands. For example, an increment command routes the CR0 outputs through the incrementer to CR0 inputs. Each clock pulse then advances the microprogram memory one location.

CR1 is used with program flow repeat cycles. The repeat count is loaded into CR1 enabling an individual instruction or sequence of instructions to be repeated until the cycle count is reached. CR2 is a general purpose register that can be used to store machine instructions or interrupt vectors. CR3 is a status or condition code register. Individual bits can be tested within the MC10801 for conditional jumps in program.

A 4-deep LIFO stack is included in the part for storing subroutine return address. A jump to subroutine command takes the present microprogram address through the incrementer and pushes it into the stack. Next Address inputs go to CR0 for the subroutine destination. A subroutine return pops the LIFO and routes the return address to CR0. A total of 16 different instructions are built into the MC10801 for various program flow requirements.

The MC10803 performs both data transfers and ALU operations. There are five I/O ports (I Bus, ϕ Bus, A Bus, D Bus, and P inputs) for transferring data to or from the part. Internally there are six storage registers. The MDR can be used to hold incoming or outgoing data. It also functions as an accumulator for the ALU. The MAR holds outgoing data for the A Bus. Four additional registers are contained in the internal register file block. A Data Matrix accepts data transfer commands and routes data between the various I/O ports and internal registers. The final block is an ALU which performs arithmetic, logic, and shift operations. Both the MC10801 and MC10803 are controlled by select lines which in turn are controlled by microprogram memory bits.

PROGRAM FLOW

Program flow charts describe the various processor operations and determine a microprogram instruction sequence. Figure 4 shows system data paths available to a programmer. These data paths, the ALU, and working registers are used in the following program flow diagrams and later to write the microprogram. All Figure 4 data paths and logic except the link bit storage are contained within the MC10803. Link bits are used with multiply and divide to hold shift in and shift out status. Available CR3 register bits in the MC10801 provide this function.

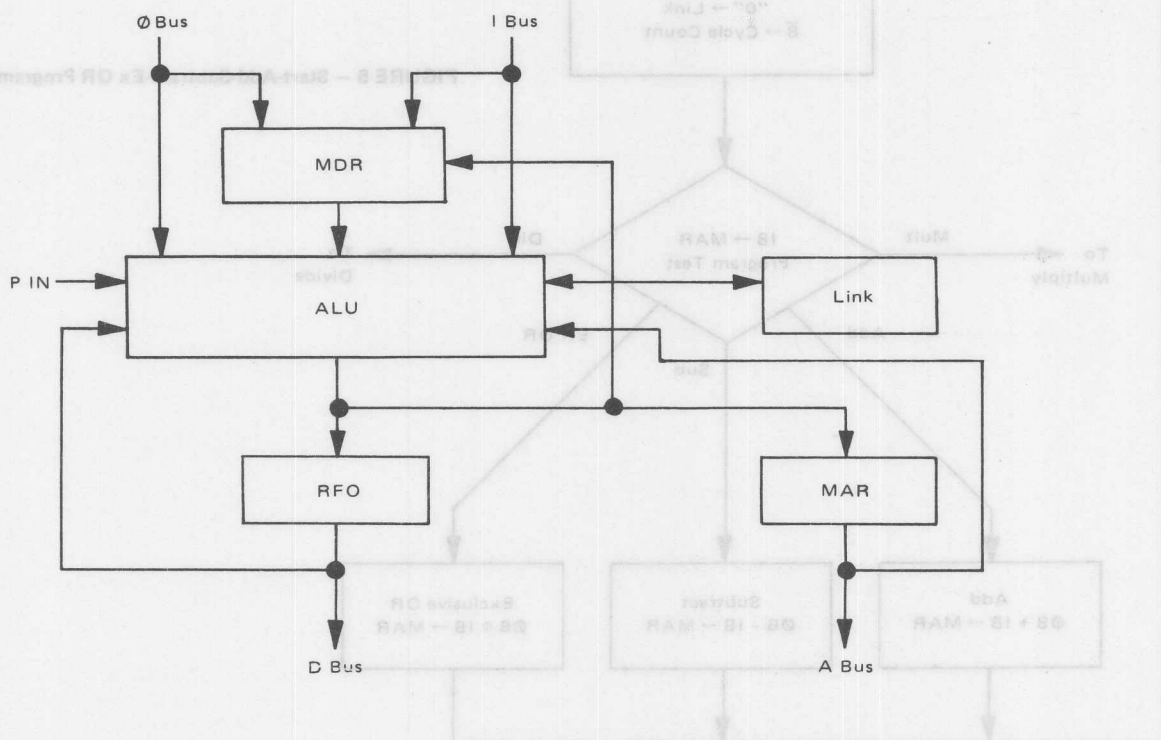


FIGURE 4 — Processor Data Paths

Figure 5 shows the flow patterns for add, subtract, and exclusive OR. Initially the processor is random on power-up or displaying the results of a previous calculation. It is in a continuous loop waiting for a start signal. After receiving start, the program performs several initialize functions. RFO is zeroed as required for add, subtract, and exclusive OR, see Figures 1 and 4. Setting up for a possible multiply or divide the link bit is set to zero, the complement of 8 is loaded in the MC10801 program cycle counter CR1, and I Bus data is transferred to the MAR. These functions are common to both multiply and divide. Location at this point in program saves microinstructions.

Program flow continues to a main decision point. Here, the processor looks at the program select inputs and picks one of five possible program directions. Add, subtract, and exclusive OR are each one step programs, Figure 5. The ALU looks at the ϕ Bus and I Bus inputs, performs the selected operation, and transfers the answer to MAR. The program jumps back to start and displays the answer.

Multiply is an implementation of Booth algorithm as shown below:

1. Load multiplier into MAR
2. Load multiplicand into MDR
3. Zero RFO and link bit
4. Set cycle counter to 8
5. Test MAR LSB and link bit

LSB	LINK	
0	0	GO TO 8
0	1	GO TO 6
1	0	GO TO 7
1	1	GO TO 8

6. Subtract RFO - MDR \rightarrow RFO, go to 8
7. Add RFO + MDR \rightarrow RFO
8. Arithmetic shift right RFO \rightarrow MAR \rightarrow LINK
9. Decrement cycle counter; if \neq zero, go to 5
10. End.

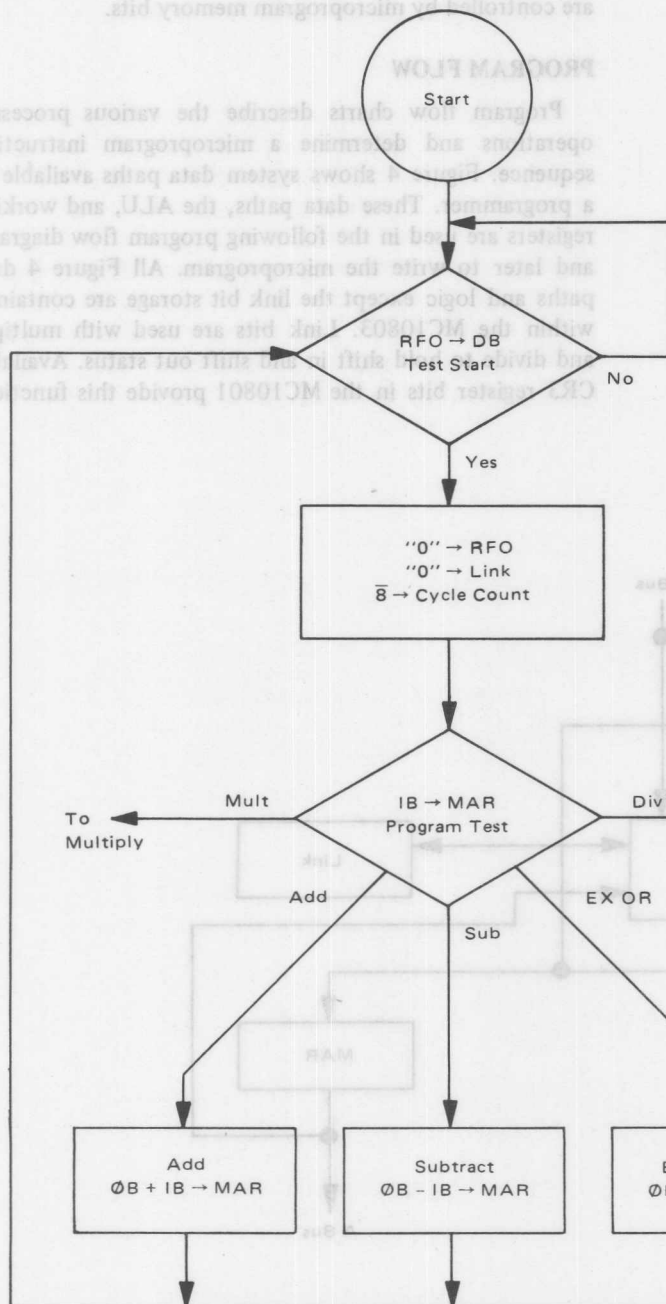
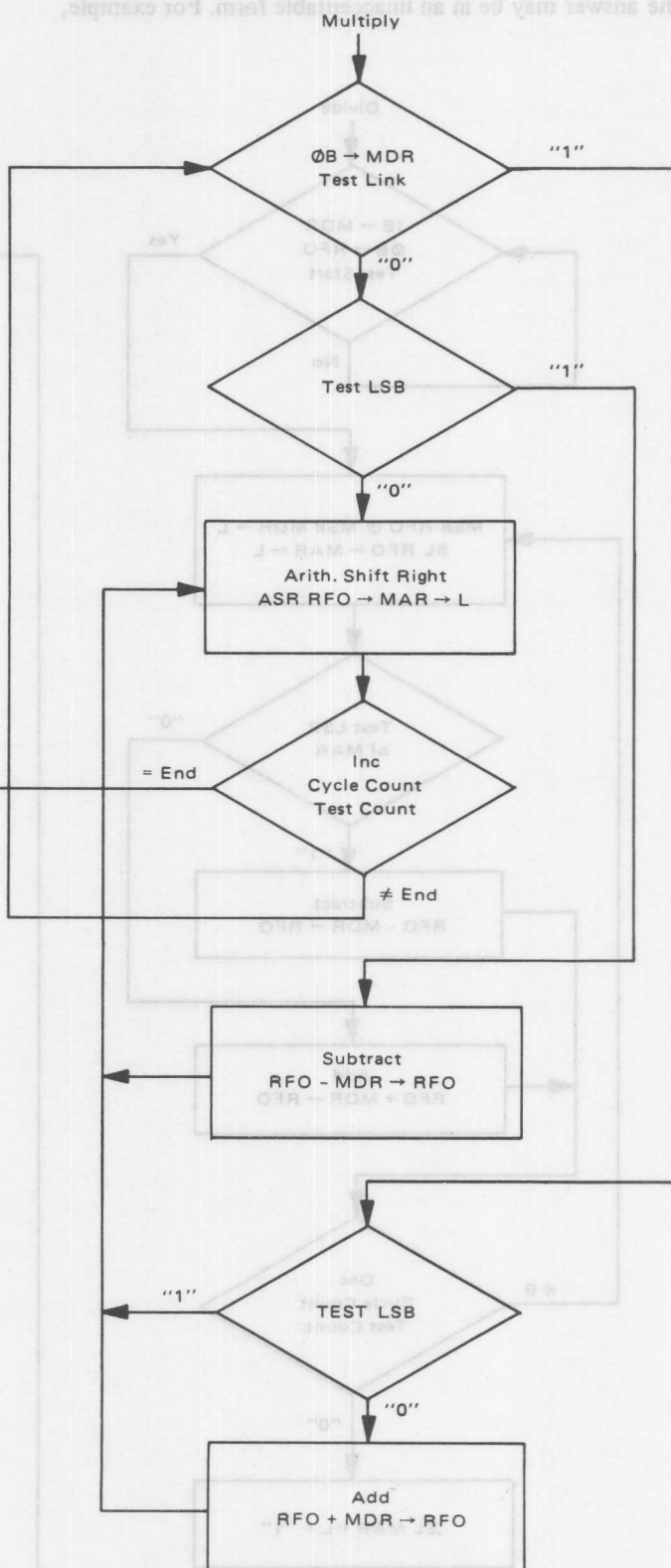


FIGURE 5 — Start-Add-Subtract-Ex OR Program Flow Diagram

The multiply flow diagram is given in Figure 6. Figure 5 sets up the multiplier, RFO, cycle counter, and link bit. Figure 6 loads the multiplicand and contains the program paths for add-shift, subtract-shift, or shift-only as required for 2's complement multiplication. As seen in the figure, the processor alternately tests the link bit and MAR LSB to select a program flow path. This simplifies the branch select compared with testing both status points in parallel. The program cycle counter is incremented after the double precision RFO/MAR shift right and tested for the end count. After 8 cycles, the program is complete. The result is a 16-bit answer with the 8 least significant bits in MAR and the 8 most significant bits in RFO.

FIGURE 6 - Multiply Flow Diagram

To
Start



Divide is an implementation of a non-restoring division algorithm as shown below:

1. Load dividend LSB into MAR
2. Load dividend MSB into RFO
3. Load divisor into MDR
4. Set cycle counter to 8
5. MSB RFO exclusive NOR MSB MDR → LINK
6. Shift left RFO ← MAR ← LINK
7. Test MAR LSB; if zero, go to 9
8. Subtract RFO - MDR → RFO, go to 10
9. Add RFO + MDR → RFO
10. Decrement cycle counter; if ≠ zero, go to 5
11. Shift right MAR, link = "1"
12. Go to format correction.

The program flow diagram starts in Figure 5 where the program cycle counter is set and the least significant bits of the dividend are transferred from the I Bus to the MAR. The remaining divide program flow is shown in Figure 7. The first block is a start loop which waits for the divisor data on the I Bus. The dividend MSB is transferred to RFO and the divisor to MDR. The program goes through eight repetitive cycles, each setting link bit status, shifting left the MAR and RFO, and performing the divisor add or subtract with RFO. After the eighth cycle MAR is shifted left with a logic "1" forced into the LSB through the link bit.

At this point, the answer is numerically correct with the quotient in MAR and the remainder in RFO. However, the answer may be in an unacceptable form. For example,

$37 \div 6$ comes out 7, remainder -5. The program flow on the right side of Figure 7 checks the format by testing $RFO - MDR = 0$, $RFO = 0$, $RFO + MDR = 0$, and MSB of $RFO = \text{MSB of dividend on } \phi \text{ Bus}$. If the format is correct, the program jumps to start. If incorrect, the quotient and remainder are corrected as shown in Figure 7.

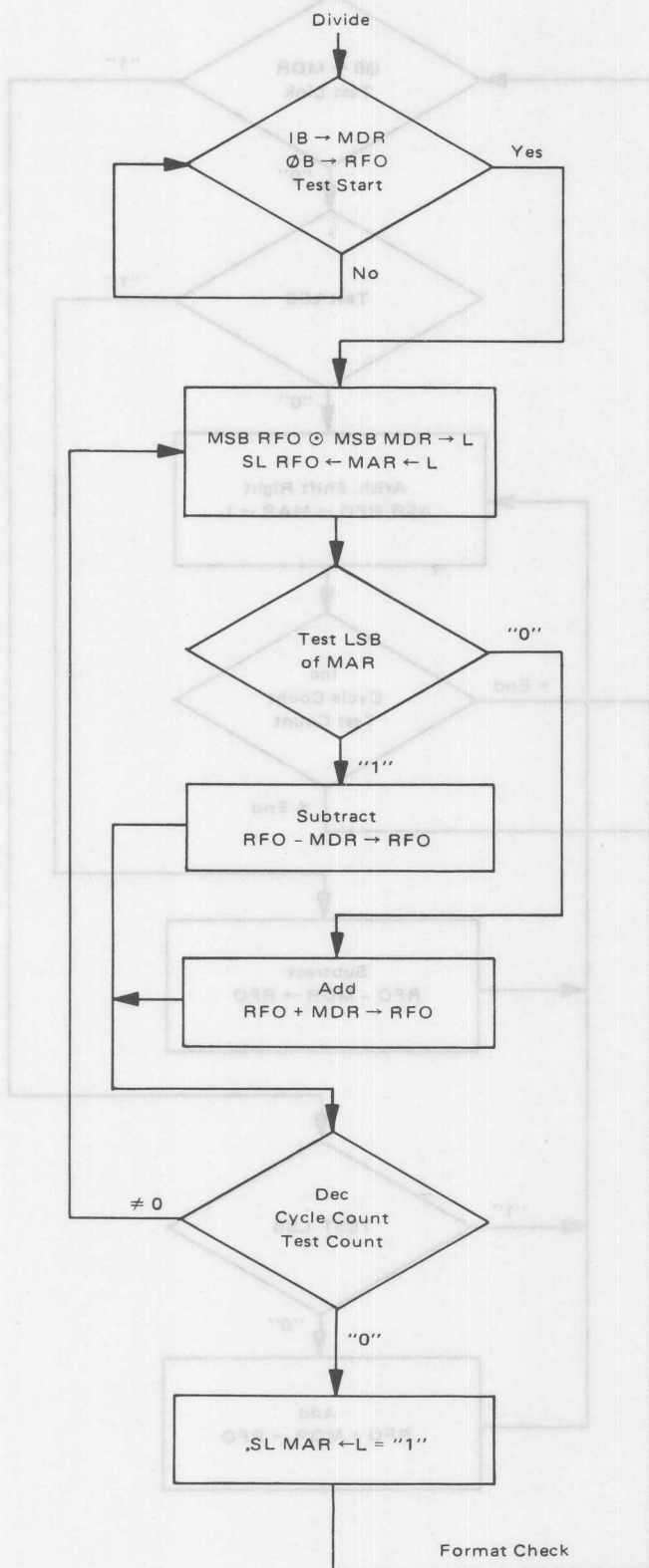
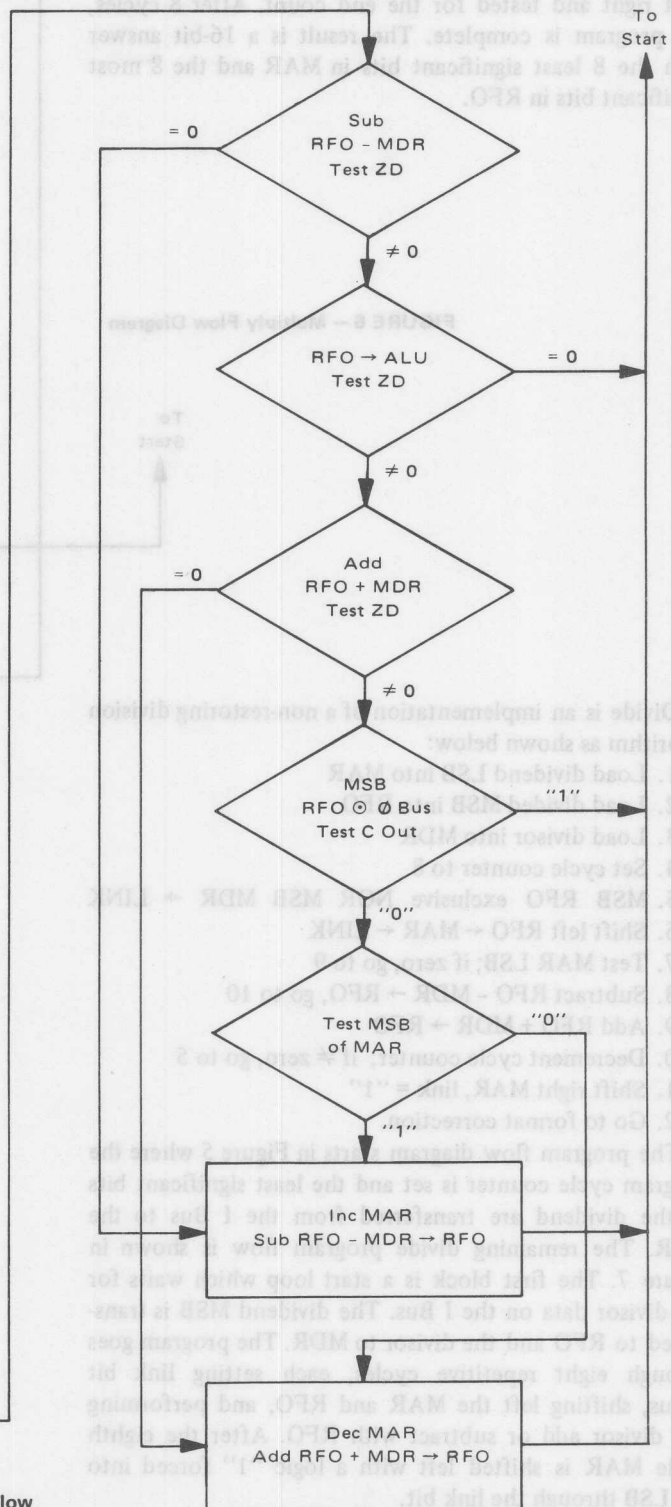


FIGURE 7 – Divide Program Flow



TIE IT ALL TOGETHER

Various test parameters can be identified from the program flow diagrams. These are 1) start, 2) program select, 3) shift right link bit, 4) MAR LSB, 5) cycle count, 6) MSB of RFO EX-NOR MDR (available on C out), and 7) zero detect. From the flow diagrams and the basic block diagram in Figure 2, the complete system is tied together as shown in Figure 8. Start and program select go directly to the MC10801. Cycle count is a feature of the MC10801 and requires no special treatment. Zero detect, MAR LSB, and C out are gated to the MC10801 as required for program test. Shift right link bit must be stored between microinstructions and uses bit 1 of register CR3. This bit can be tested internal to the MC10801 for program flow decisions.

Register CR3 holds three program status bits. CR3, bit 0, is a page address representing the fifth microprogram memory address bit. CR3, bit 1, is the shift right link bit and can be gated to the ALU C out (shift right input). CR3, bit 2 performs the same function for shift left and can be gated to ALU C in.

The Figure 8 microprogram memory fields have been refined from Figure 2. Individual bits control the program test inputs, shift link bits, C in, and MC10803 P inputs. The three LSI parts have been previously defined. The remaining blocks in Figure 8 are SSI. A dual flip-flop (10131) supplies bounce elimination and start signal pulse shaping. Leftover NAND and OR gates are used for power-up reset and a crystal oscillator clock.

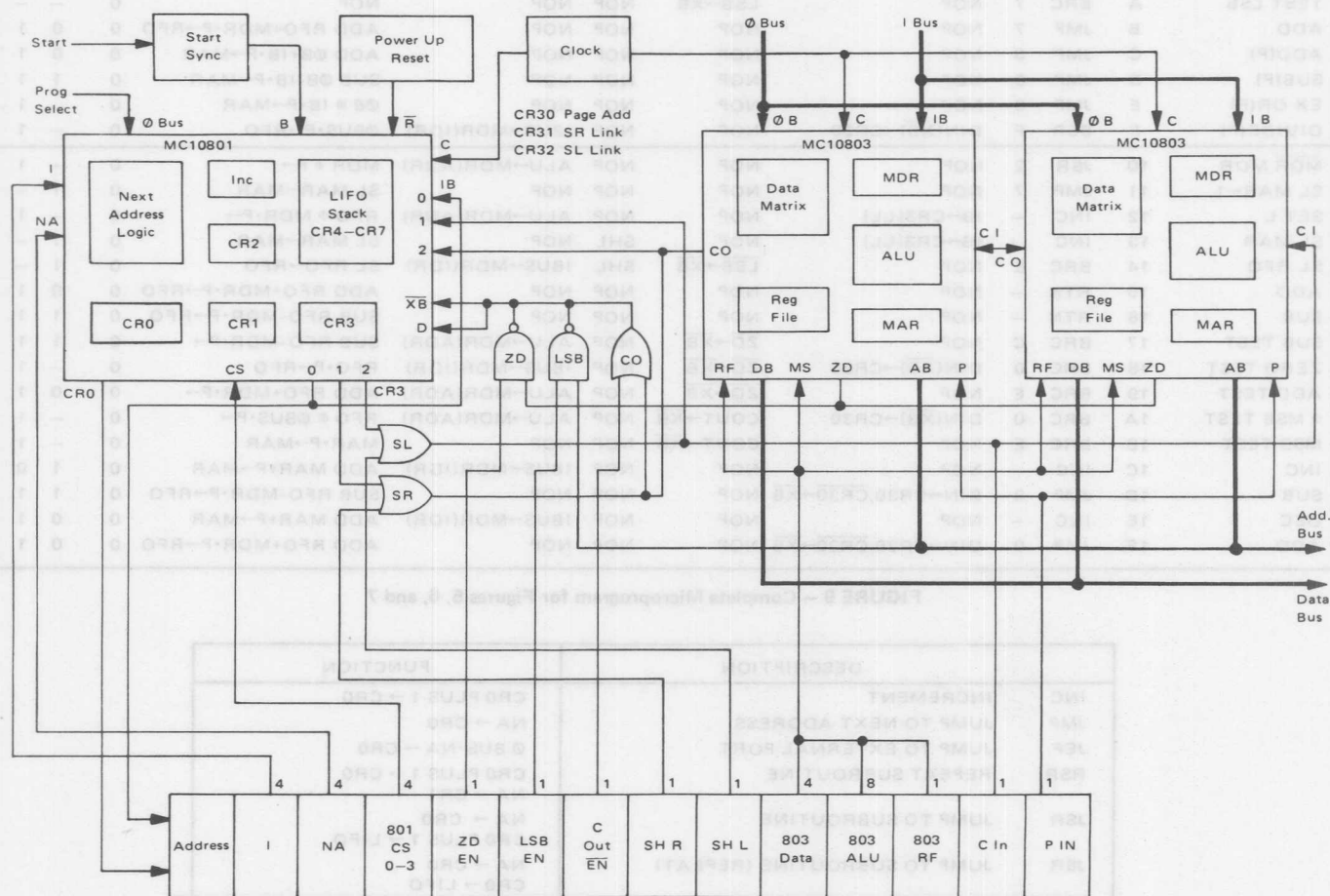


FIGURE 8 - Processor Logic Diagram

MICROPROGRAM THE MEMORY

The complete microprogram for Figures 5, 6, and 7 is given in Figure 9. A function column briefly describes each program step. Add column is the microprogram word address in hexadecimal format. Table 1 shows MC10801 sequencing instructions. INC and JMP are direct transfers of a new address to the microprogram address register, CR0. JEP is used for the five-way program select jump. Program select inputs are connected to the MC10801 ϕ Bus port and program select information

corresponds to the individual program starting address. RSR loads the repeat cycle count into CR1. JSR jumps to the NA input address for a subroutine location and pushes a return address into the LIFO stack. If the subroutine is to be repeated, CR0 is pushed into the stack. Otherwise, CR0 plus 1 is loaded in the stack. This repeat function is automatically controlled by the cycle counter internal to the MC10801. RTN pops the LIFO stack into CR0 for a subroutine return. In a repeat mode, the

FUNCTION	ADD	I	NA	801 CR3	\overline{XB}	SH	DATA	ALU	RF	CIN	P
READ	0	BSR	0	NOP	NOP	NOP	RFO \rightarrow DB (FDB)	NOP	0	—	—
"0" RFO,L	1	RSR	8	IB \rightarrow CR3(L)	NOP	NOP	IBUS \rightarrow MDR (IDR)	RFO \cdot P \rightarrow RFO	0	0	0
PROG TEST	2	JEP	F	NOP	NOP	NOP	NOP	MDR \cdot P \rightarrow MAR	0	—	1
MULTIPLY(P)	3	JSR	5	NOP	NOP	NOP	\emptyset BUS \rightarrow MDR (\emptyset DR)	NOP	0	—	—
END	4	JMP	0	NOP	NOP	NOP	NOP	NOP	0	—	—
TEST LINK	5	BRC	A	CR3 \rightarrow \overline{XB} (L)	NOP	NOP	NOP	NOP	0	—	—
TEST LSB	6	BRC	9	NOP	LSB \rightarrow \overline{XB}	NOP	NOP	NOP	0	—	—
SHR MSB	7	INC	—	IB \rightarrow CR3(L)	NOP	NOP	NOP	ASR RFO \rightarrow RFO	0	1	—
SHR LSB	8	RTN	—	IB \rightarrow CR3(L)	NOP	SHR	NOP	LSR MAR \rightarrow MAR	0	1	—
SUB	9	JMP	7	NOP	NOP	NOP	NOP	SUB RFO-MDR \cdot P \rightarrow RFO	0	1	1
TEST LSB	A	BRC	7	NOP	LSB \rightarrow \overline{XB}	NOP	NOP	NOP	0	—	—
ADD	B	JMP	7	NOP	NOP	NOP	NOP	ADD RFO+MDR \cdot P \rightarrow RFO	0	0	1
ADD(P)	C	JMP	0	NOP	NOP	NOP	NOP	ADD \emptyset B+IB \cdot P \rightarrow MAR	0	0	1
SUB(P)	D	JMP	0	NOP	NOP	NOP	NOP	SUB \emptyset B-IB \cdot P \rightarrow MAR	0	1	1
EX OR(P)	E	JMP	0	NOP	NOP	NOP	NOP	\emptyset B \oplus IB \cdot P \rightarrow MAR	0	—	1
DIVIDE(P)	F	BSR	F	DIN(\overline{XB}) \rightarrow CR30	NOP	NOP	IBUS \rightarrow MDR (IDR)	\emptyset BUS \cdot P \rightarrow RFO	0	—	1
MDR MDR	10	JSR	2	NOP	NOP	NOP	ALU \rightarrow MDR (ADR)	MDR \oplus P \rightarrow	0	—	1
SL MAR \leftarrow 1	11	JMP	7	NOP	NOP	NOP	NOP	SL MAR \rightarrow MAR	0	1	—
SET L	12	INC	—	IB \rightarrow CR3(LL)	NOP	NOP	ALU \rightarrow MDR (ADR)	RFO \oplus MDR \cdot P \rightarrow	0	—	1
SL MAR	13	INC	—	IB \rightarrow CR3(LL)	NOP	SHL	NOP	SL MAR \rightarrow MAR	0	1	—
SL RFO	14	BRC	6	NOP	LSB \rightarrow \overline{XB}	SHL	IBUS \rightarrow MDR (IDR)	SL RFO \rightarrow RFO	0	1	—
ADD	15	RTN	—	NOP	NOP	NOP	NOP	ADD RFO+MDR \cdot P \rightarrow RFO	0	0	1
SUB	16	RTN	—	NOP	NOP	NOP	NOP	SUB RFO-MDR \cdot P \rightarrow RFO	0	1	1
SUB TEST	17	BRC	C	NOP	ZD \rightarrow \overline{XB}	NOP	ALU \rightarrow MDR (ADR)	SUB RFO-MDR \cdot P \rightarrow	0	1	1
ZERO TEST	18	BRC	0	DIN(\overline{XB}) \rightarrow CR30	ZD \rightarrow \overline{XB}	NOP	IBUS \rightarrow MDR (IDR)	RFO \cdot P \rightarrow RFO	0	—	1
ADD TEST	19	BRC	E	NOP	ZD \rightarrow \overline{XB}	NOP	ALU \rightarrow MDR (ADR)	ADD RFO+MDR \cdot P \rightarrow	0	0	1
\oplus MSB TEST	1A	BRC	0	DIN(\overline{XB}) \rightarrow CR30	COUT \rightarrow \overline{XB}	NOP	ALU \rightarrow MDR (ADR)	RFO \oplus \emptyset BUS \cdot P \rightarrow	0	—	1
MSB TEST	1B	BRC	E	NOP	COUT \rightarrow \overline{XB}	NOP	NOP	MAR \cdot P \rightarrow MAR	0	—	1
INC	1C	INC	—	NOP	NOP	NOP	IBUS \rightarrow MDR (IDR)	ADD MAR+P \rightarrow MAR	0	1	0
SUB	1D	JMP	0	DIN \rightarrow CR30,CR30 \rightarrow \overline{XB}	NOP	NOP	NOP	SUB RFO-MDR \cdot P \rightarrow RFO	0	1	1
DEC	1E	INC	—	NOP	NOP	NOP	IBUS \rightarrow MDR (IDR)	ADD MAR+P \rightarrow MAR	0	0	1
ADD	1F	JMP	0	DIN \rightarrow CR30,CR30 \rightarrow \overline{XB}	NOP	NOP	NOP	ADD RFO+MDR \cdot P \rightarrow RFO	0	0	1

FIGURE 9 — Complete Microprogram for Figures 5, 6, and 7

	DESCRIPTION	FUNCTION
INC	INCREMENT	CR0 PLUS 1 \rightarrow CR0
JMP	JUMP TO NEXT ADDRESS	NA \rightarrow CR0
JEP	JUMP TO EXTERNAL PORT	\emptyset BUS \cdot NA \rightarrow CR0
RSR	REPEAT SUBROUTINE	CR0 PLUS 1 \rightarrow CR0 NA \rightarrow CR1
JSR	JUMP TO SUBROUTINE	NA \rightarrow CR0 CR0 PLUS 1 \rightarrow LIFO
JSR	JUMP TO SUBROUTINE (REPEAT)	NA \rightarrow CR0 CR0 \rightarrow LIFO
RTN	RETURN FROM SUBROUTINE	LIFO \rightarrow CR0
RTN	RETURN FROM SUBROUTINE (REPEAT)	LIFO \rightarrow CR0 CR1 PLUS 1 \rightarrow CR1
BRC	BRANCH ON CONDITION	CR0 PLUS 1 \rightarrow CR0 (TEST = 0) NA \rightarrow CR0 (TEST = 1)
BSR	BRANCH TO SUBROUTINE	CR0 PLUS 1 \rightarrow CR0 (START) NA \rightarrow CR0 (START)

TABLE 1
Microprogram Flow Commands

cycle counter CR1 is automatically incremented on this instruction. BRC looks at the selected test parameter and depending on test status executes either an increment or jump to NA inputs. BSR, normally a conditional branch to subroutine, is used to enable the MC10801 Branch input for a jump on start signal instruction.

The NA column in Figure 9 is a jump destination. It is used with all flow instructions except INC and RTN which require no additional information for program flow. The NA field in this system is four bits wide and operates within a 16-word memory page. Page addressing is part of the 801 CR3 column in Figure 9.

The MC10801 CR3 register holds the memory page address and two shift link bits. This register is parallel loaded on an I Bus to CR3 command, see Table 2. Notice CR3 bit 0 is connected to IB0 in Figure 8. This holds the page address constant on the I Bus parallel load command. The multiply program requires testing status of the shift right link bit for a program flow decision. Testing is accomplished by gating CR3 bit 1 to \overline{XB} and making a program flow decision with a BRC instruction. The last two Table 2 commands control page address. For conditional jumps between pages CR3 bit 0 is loaded from D In connected to \overline{XB} . Alternately, CR3 bit 0 can be toggled for unconditional jumps.

The remaining fields in Figure 9 are relatively straightforward. \overline{XB} can be selectively programmed to zero detect, LSB, or C Out. These are used with the BRC program flow instruction to make decisions in program. The SH R field gates the shift right link bit in MC10801 CR3 bit 1 to carry out of the MC10803. It is disabled for all other arithmetic functions. The SH L field performs a similar

function for the shift left link bit. Additional information on shift operations follows in the paragraph on MC10803 ALU operations.

The data field selects four different MC10803 data transfer functions. FDB transfers the information in RFO to the D Bus for answer display. \emptyset DR and IDR read the processor inputs and transfer data to the MDR accumulator. ADR routes the ALU output to MDR. This is used to modify the MDR contents as in Figure 9, word 10, or to avoid changing information in RFO or MAR (program words 12, 17, 19, and 1A).

The MC10803 ALU selects between RFO, MAR, MDR, \emptyset Bus, I Bus, and P for operands. AND and exclusive OR are selected logic functions, with add and subtract selected for arithmetic. These functions combine with the P inputs for special operations: ALU = zero, word 1; MDR invert, word 10; and MAR decrement, word 1E. Five different shift combinations are formed with the ALU, MC10801 CR3, shift, and C In fields, as shown in Table 3. An IB \rightarrow CR3 in the MC10801 CR3 field connects the shift out to a link bit. The shift field routes link bit to the shift input. Word 11 disables the shift left link allowing C In to become the shift input. Word 13 uses the shift left link for a rotate.

Only selected MC10801 and MC10803 functions have been described as required for this program. They are a small percent of the total combinations available to a system designer. The LSI circuits therefore adapt to a wide range of system architectures and applications. Additional information is available on component data sheets.

FUNCTION	DESCRIPTION
IB \rightarrow CR3	PARALLEL LOAD CR3
CR31 \rightarrow \overline{XB}	TEST SHIFT RIGHT LINK
DIN \rightarrow CR30	CHANGE PAGE ADDRESS FROM D IN
DIN \rightarrow CR30, $\overline{CR30} \rightarrow \overline{XB}$	TOGGLE PAGE ADDRESS

TABLE 2
MC10801 CR3 Commands

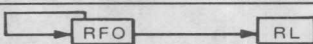
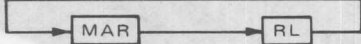
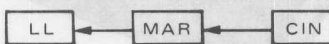
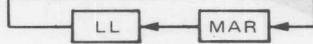
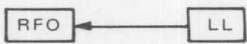
FUNCTION	OPERATION
ASR RFO \rightarrow RFO	
LSR MAR \rightarrow MAR	
SL MAR \rightarrow MAR	
SL MAR \rightarrow MAR	
SL RFO \rightarrow RFO	

TABLE 3
ALU Shift Commands

THE FINAL SYSTEM

Figure 10 is a picture of the complete wirewrapped system. The three LSI parts are in 48-pin quad-in-line packages with memories and SSI in standard 16-pin packages. IC headers hold discrete resistors and capacitors for the crystal oscillator, start pulse shaper, and power-up reset. The four remaining packages are pull-down resistors as required for ECL signal lines. A 10 MHz clock (100 ns microinstruction time) gives program execution times of 500 ns for add, subtract, and exclusive OR. The longest multiply is 5.3 μ s and the longest divide is 5.2 μ s. Additional circuits used as pipeline registers would reduce the microinstruction time, but the goal of this project is to keep part count and cost down. System power is under 14 watts including drive for 16 LEDs used to display the answer.



FIGURE 10 — Complete Wirewrapped System

Are bipolar LSI processors fast? This system is designed with the industry's fastest bipolar LSI circuits, the M10800 family, but does not fully utilize the speed potential to minimize part count. Even so, the system performs the 8-bit 2's complement multiplication approximately 100 times faster than a 1.0 μ s MOS microprocessor, 10 times speed improvement is gained with clock time, and 10 times speed improvement from architecture and microprogramming advantages.



MOTOROLA Semiconductor Products Inc.